



# Kafka Monitoring: What Matters!

Amrit Sarkar

THIS IS NOT A CONTRIBUTION

# Agenda

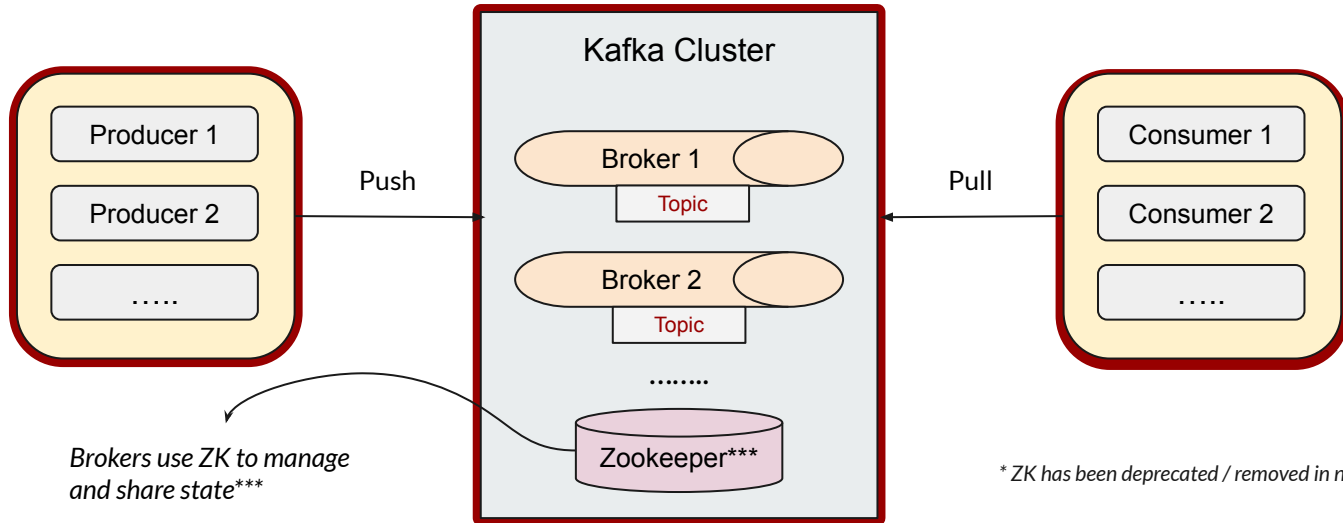


- Kafka Basics
- Performance Areas
- Need for Observability
  - Monitoring Options
- Performance classification around Components
- Kafka Consumer Lag evaluation
  - absolute to relative
- Trend Analysis

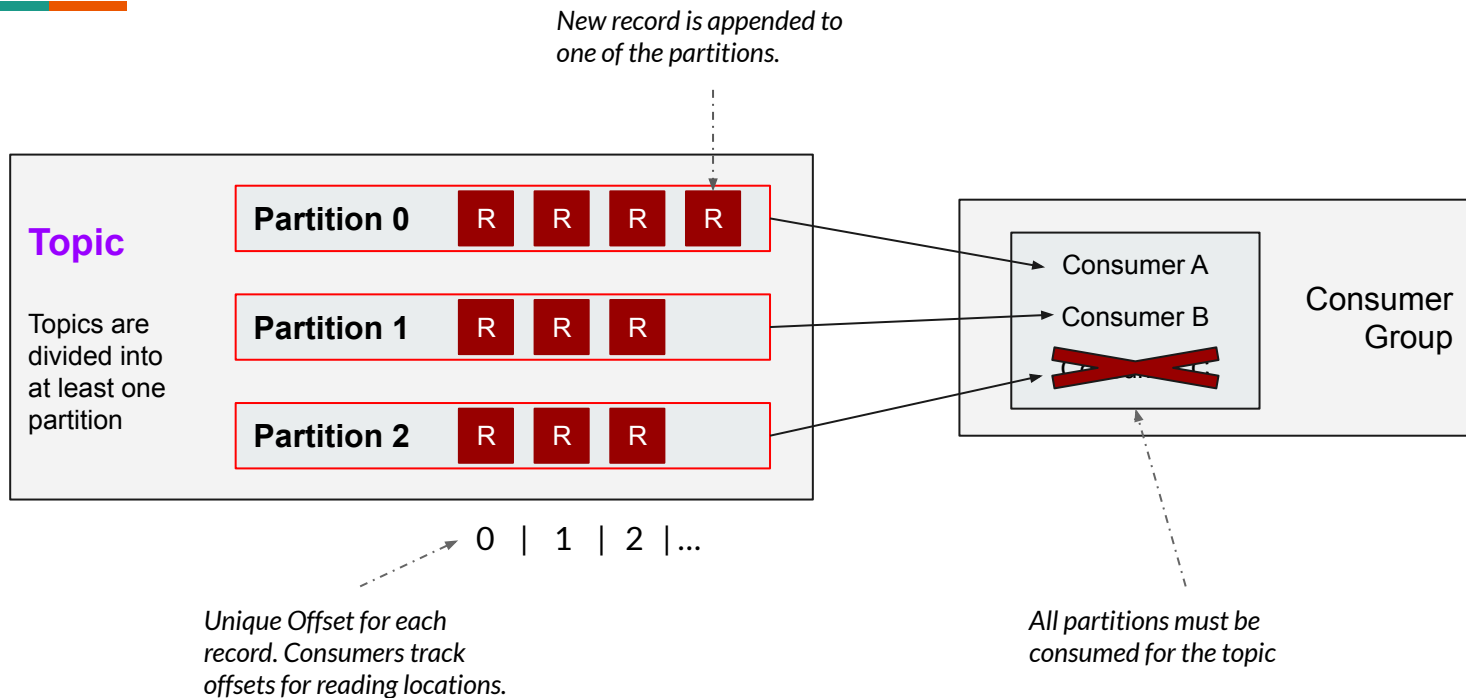
# Kafka Basics

Kafka moves data between producers (writers) and consumers (readers),  
with data protection, high availability, low latency at high scale!

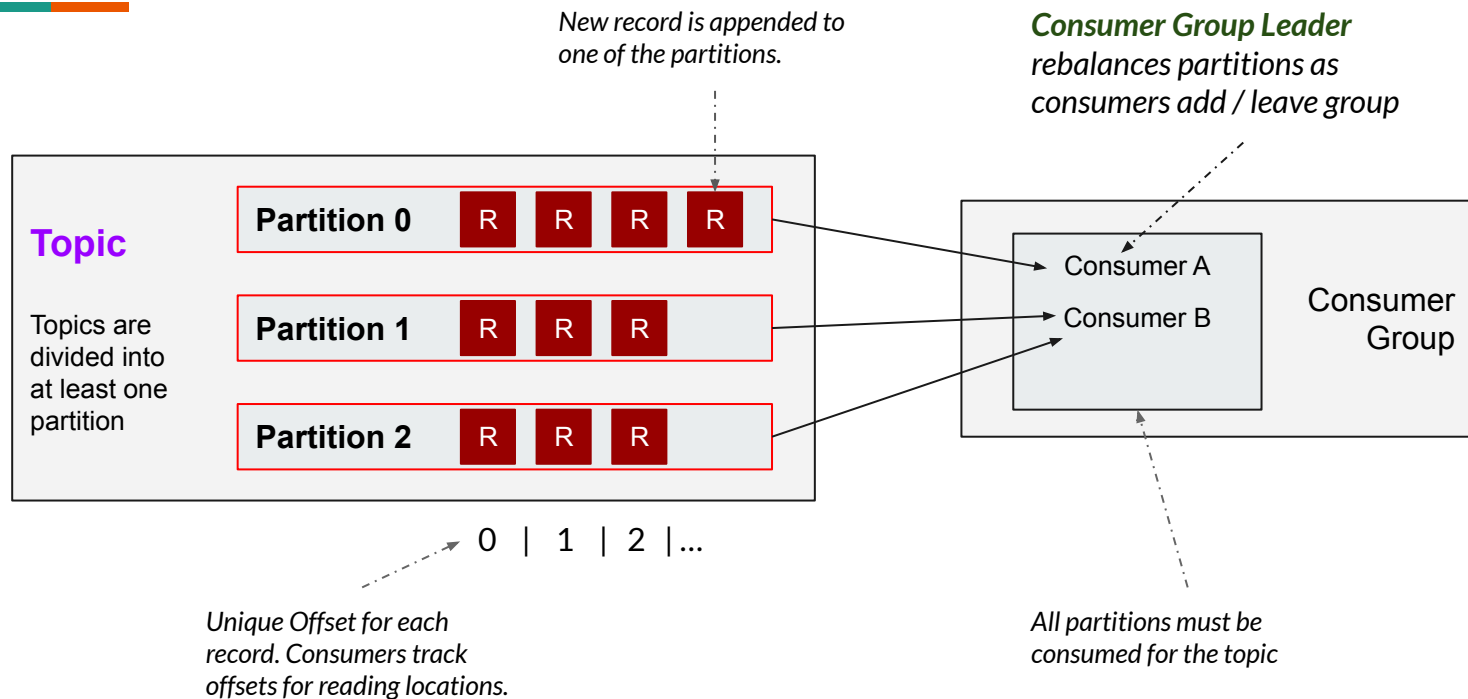
**Use cases:** Metrics, Log Aggregation Solution & Stream Processing



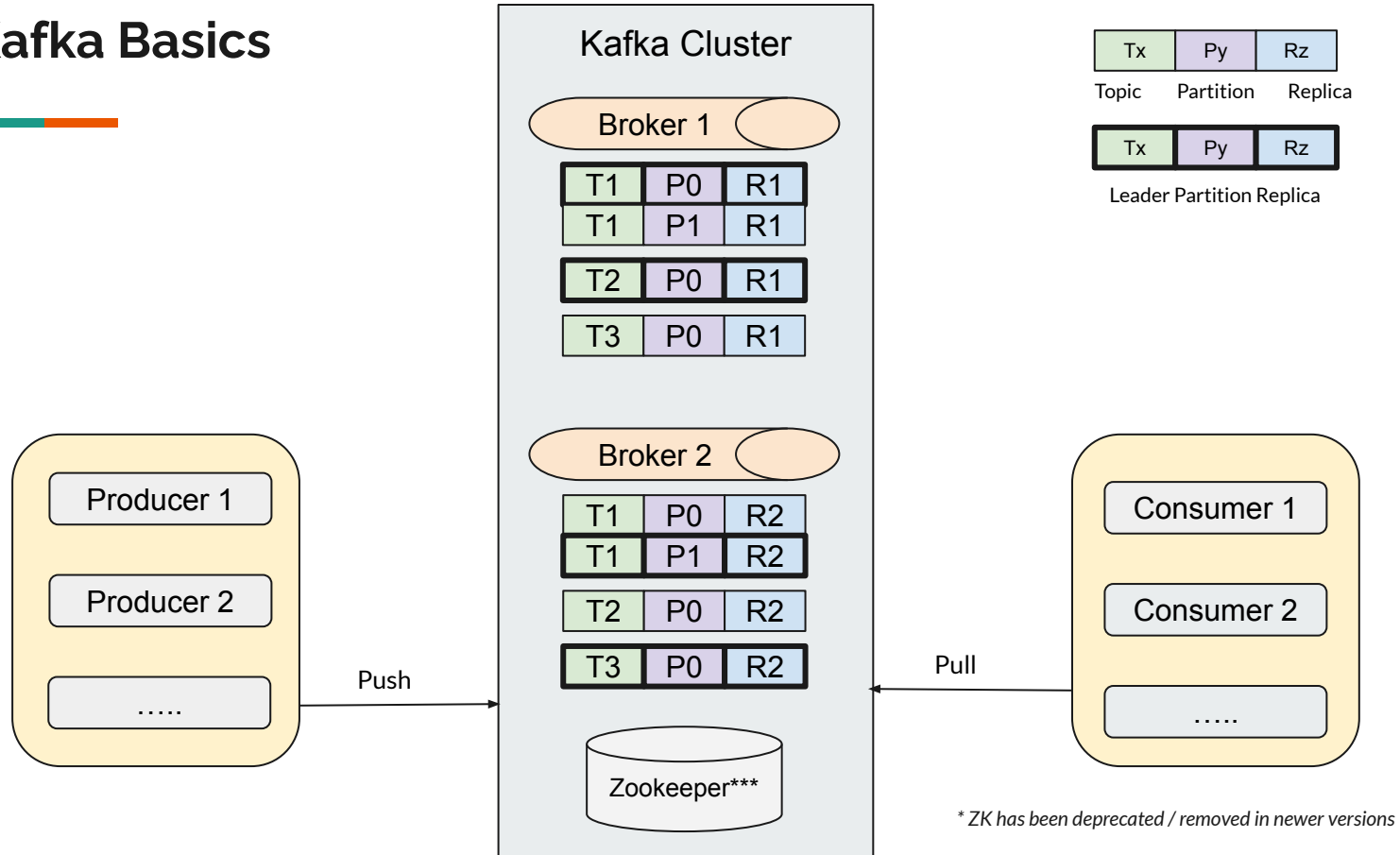
# Kafka Basics



# Kafka Basics



# Kafka Basics

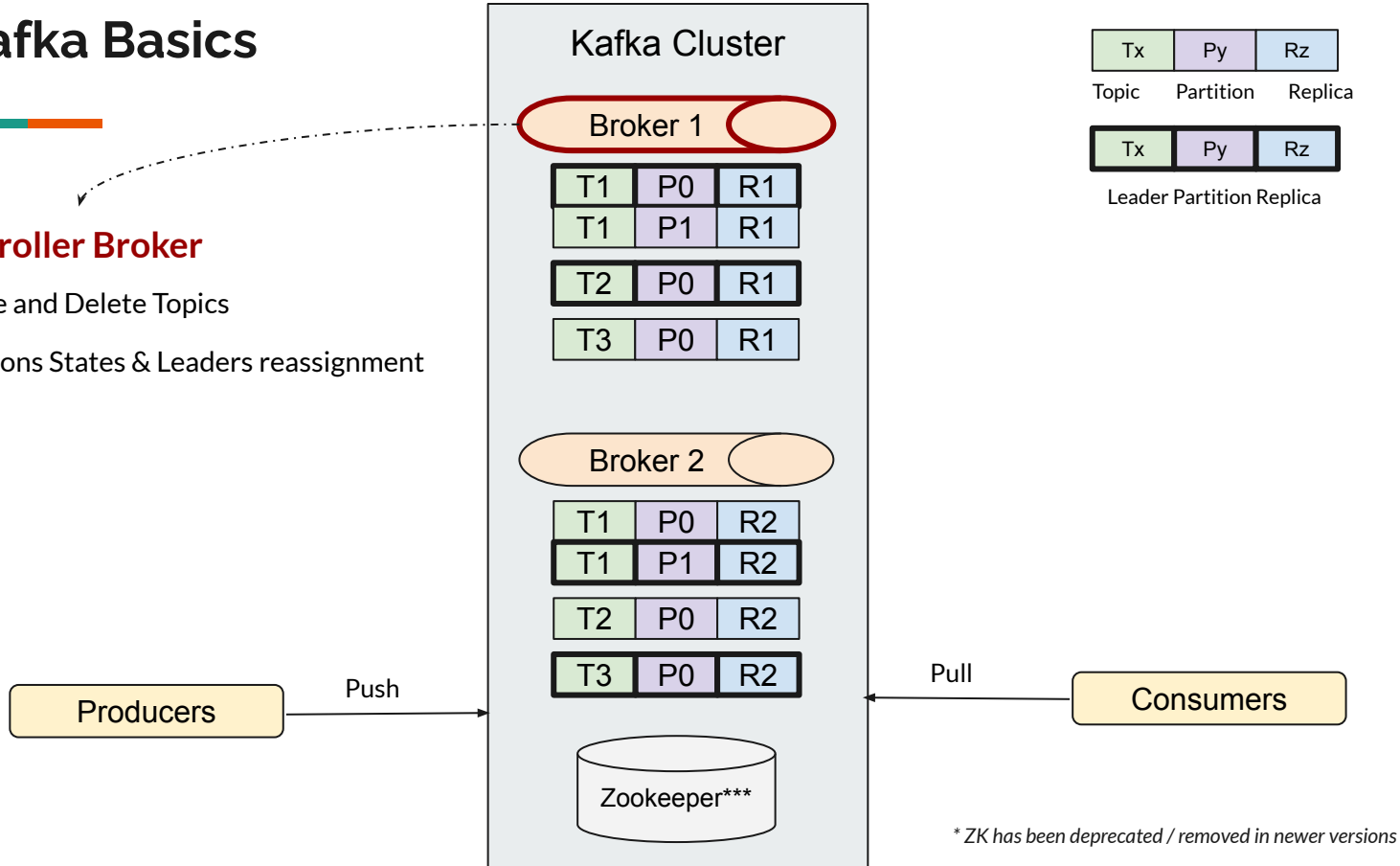


# Kafka Basics



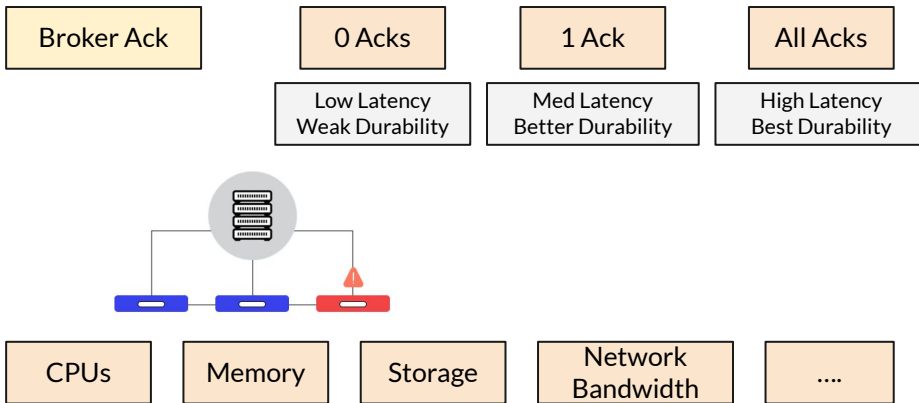
## Controller Broker

- \* Create and Delete Topics
- \* Partitions States & Leaders reassignment



# Performance Areas

- Throughput & Latency
  - Production Rates
  - Consumption Rates
  - Consumers' Lag
- Data Integrity
  - Reads Confirmation
  - Writes Confirmation
- Fault Tolerance
  - No business impact on failure
- Resource Usage

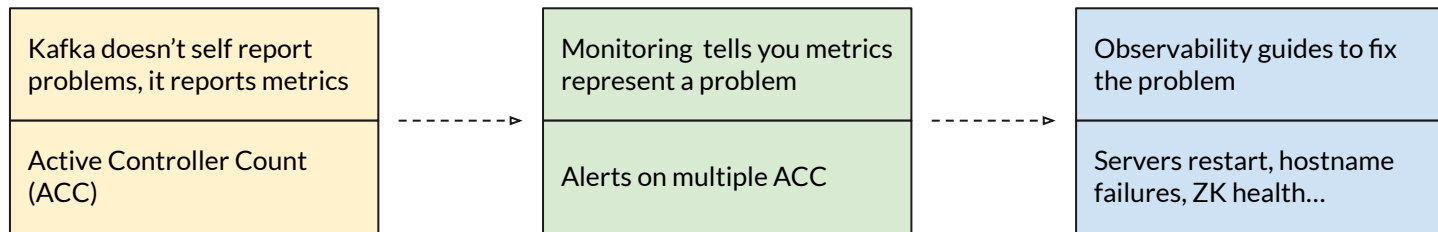




# Why do we need Observability?

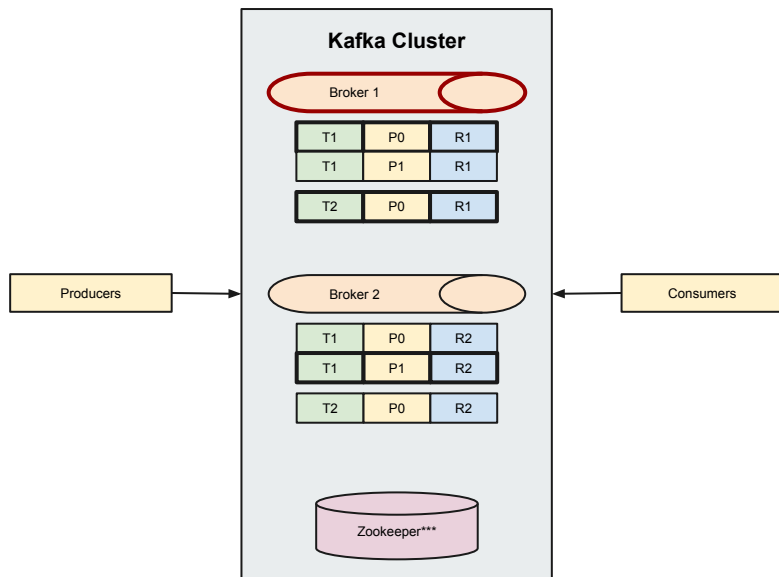
Pre-built dashboards monitor and alert for anticipated future performance issues.

Explore and quickly identify unanticipated issue root causes in an observability scenario.



# Performance classification around Components

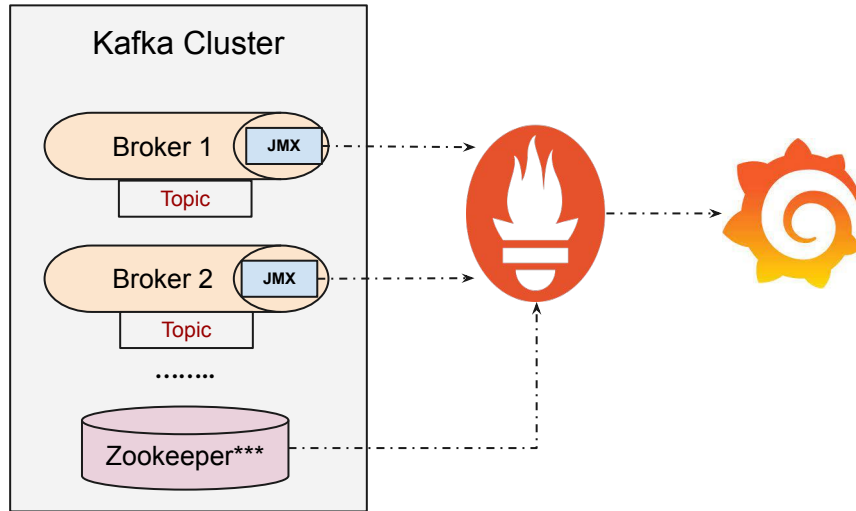
Each component act as potential factor in the performance of Kafka messaging system.



\* ZK has been deprecated / removed in newer versions

- Producers
  - Rate
- Brokers
  - Topics Health
  - Load Distribution
- Topics
  - Partition Health
  - Load Distribution
- Consumers
  - Rate
  - **Lag**
- Generic
  - Transmission (Network) health
  - Capacity

# Monitoring Options - Getting Metrics In



- Confluent Control Center
- KafDrop
- Yahoo Kafka Manager
- Cruise Control
- Kafka Monitor
- Kafka Tool

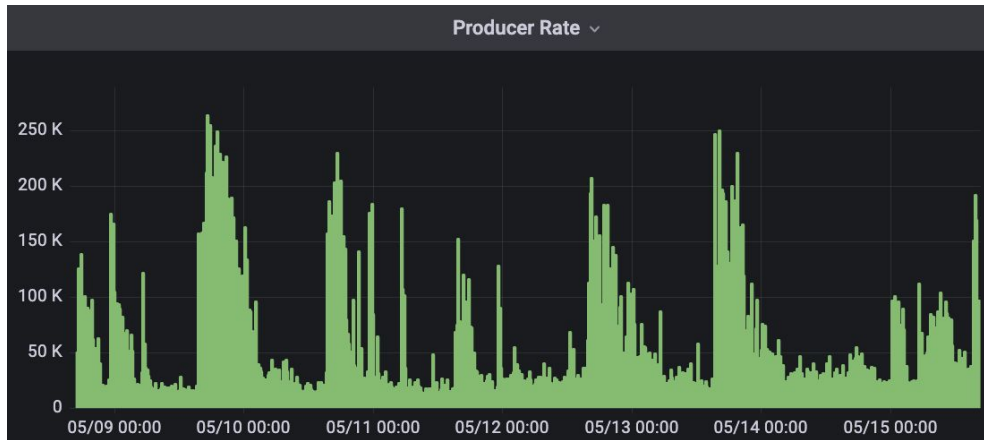
... and more

# Producer: Rate



- **send()** function call (or similar) to push data → [RecordMetadata](#) object
  - **offset()** function returns a **LONG** → offset of the the record in the topic-partition.

*Offset value can be pushed to metrics store for visualisation.*



# Producer: Compression & Latency



- Bigger batches →
  - higher throughput
  - less compression
  - Small enough to keep GC ⚡  
( $\ll 10\text{mb}$ )

*Ideally*

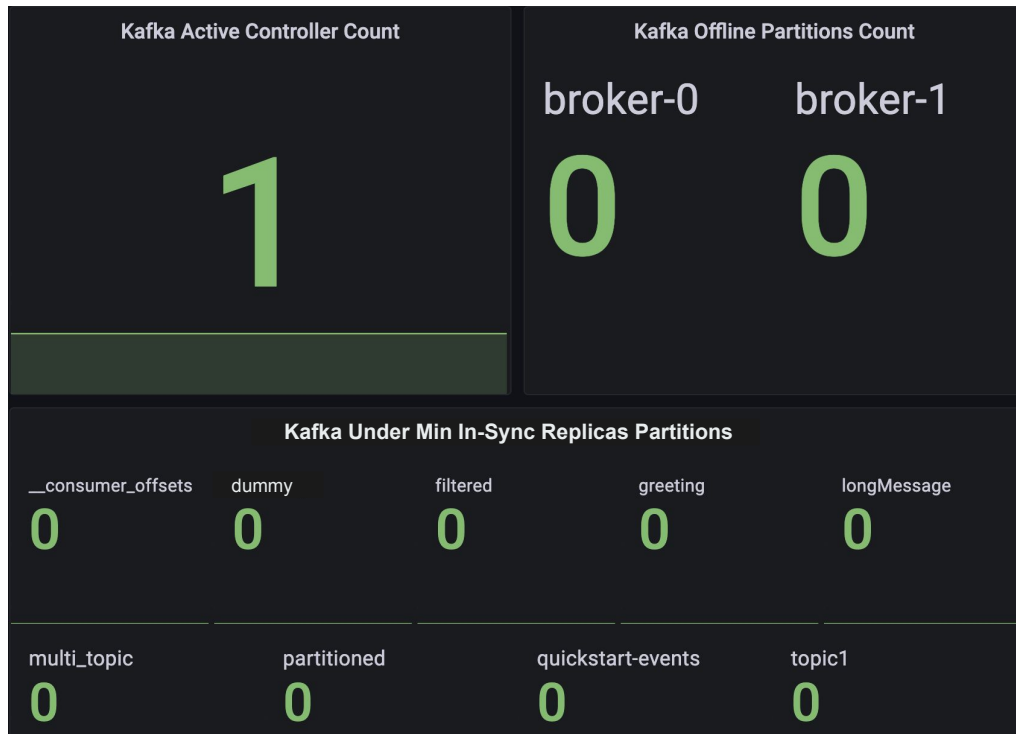
- Batch Size in Bytes → **Optimally High**  
*kafka.producer:type=producer-metrics,client-id="{client-id}"  
batch-size-avg*
- Compression Rate → **LOW**  
*kafka.producer:type=producer-metrics,client-id="{client-id}"  
compression-rate-avg*
- Request Latency → **LOW**  
*kafka.producer:type=producer-metrics,client-id="{client-id}"  
request-latency-avg*

# The Big Four - Key Metrics (JMX)



- Number of active controllers, **must be = 1**  
*kafka.controller:name=ActiveControllerCount,  
type=KafkaController*
- Number of under min ISR partitions, **must be = 0**  
*kafka.server:name=UnderMinIsrPartitionCount,  
type=ReplicaManager*  
*Checkout: 'UnderReplicatedPartitions' metric too*
- Number of offline partitions, **must be = 0**  
*kafka.controller:name=OfflinePartitionsCount,  
type=KafkaController*
- Consumer Lag (per partition)  
*kafka.consumer:name=MaxLag,  
type=ConsumerFetcherManager,clientId={[-.w]+}*

# The Big Four - Key Metrics (JMX)



# Brokers' Health

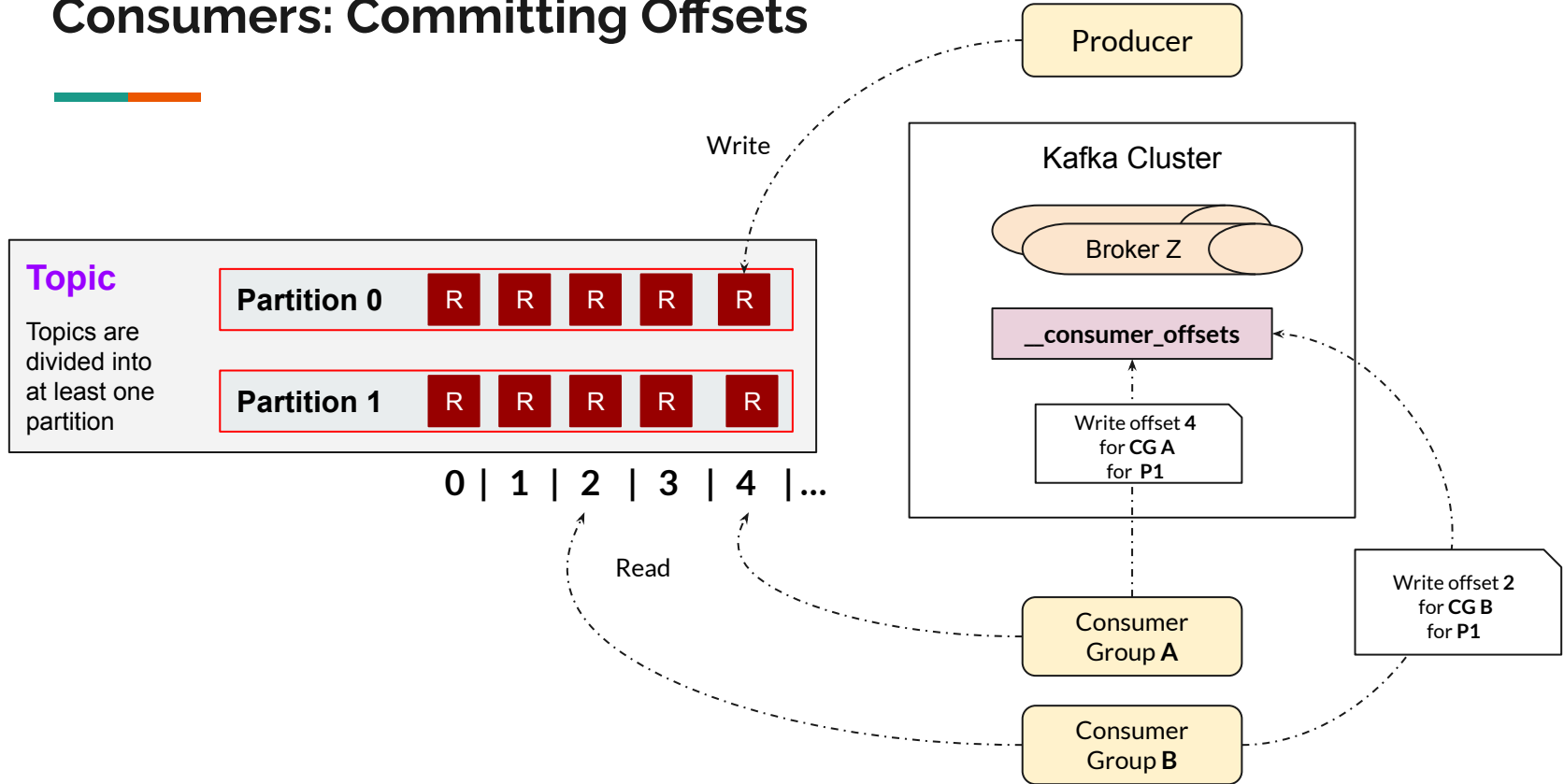
- Load Skewness (number of partitions on a broker)  
*kafka.server:name=PartitionCount,  
type=ReplicaManager*
- Log Flush Latency  
*kafka.log:name=LogFlushRateAndTimeMs,  
type=LogFlushStats*
- Network Request & Error Rate
- Fetcher Lag (per topic per partition)  
*kafka.server:name=ConsumerLag,  
type=FetcherLagMetrics,clientId=([-\.w]+),topic=([-\.w]+),  
partition=([0-9]+)*



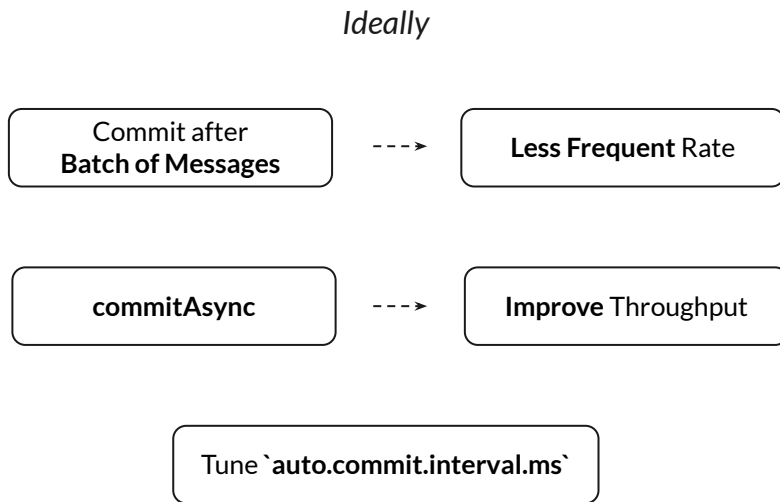
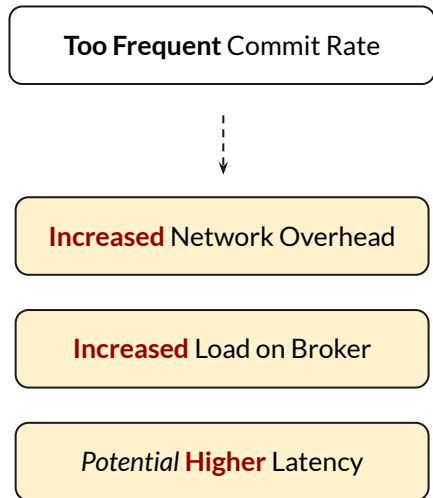
# Brokers' Health



# Consumers: Committing Offsets



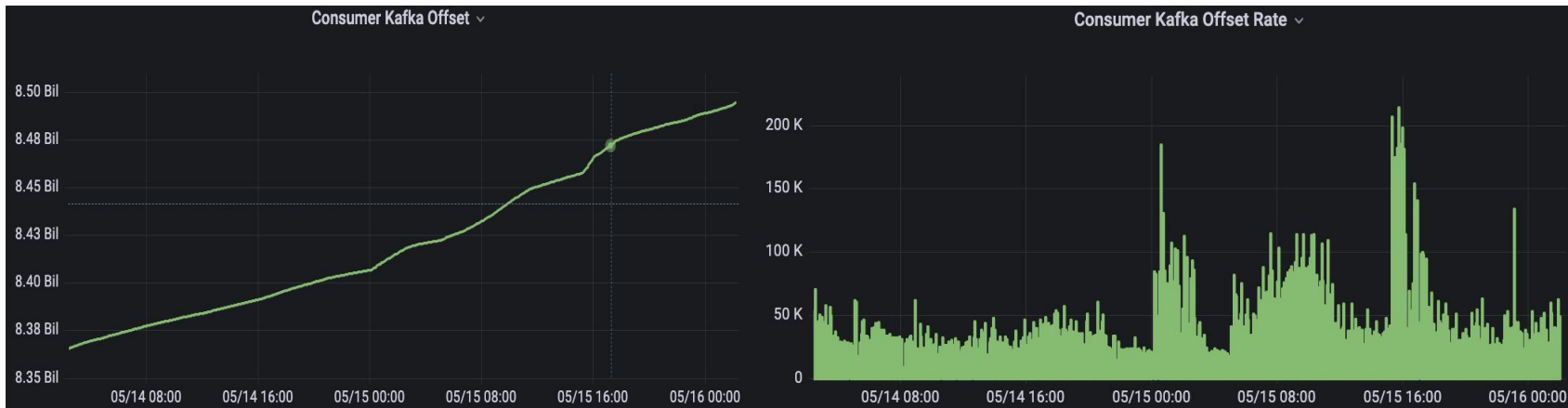
# Consumers: Commit Rate



# Consumption Rate

`__consumer_offsets` topic can be consumed;  
offset long value emitted as metric to visualise **every consumer-partition's committed offset** in near real-time.

Reference [Burrow](#) (discussed later) which already does it.



# Lag, Evaluations & Alerts: Burrow

Monitoring tool provides **consumer lag check as a service**

Exposes **offset lag** for all consumer-partition combination as **Prometheus metrics**.



**Monitors committed offsets** and **calculates the status of those consumers** on demand.

- Able to send alerts

# Lag - Offsets Trend Evaluation

**Lag = Head Offset of the Broker - Consumer's Offset**

Store metrics locally in TSDB format. Evaluation runs on a sliding window periodically.



Status of consumer: OK

Lag Series with no Uptrend & Consumer Offset Series not Stalled

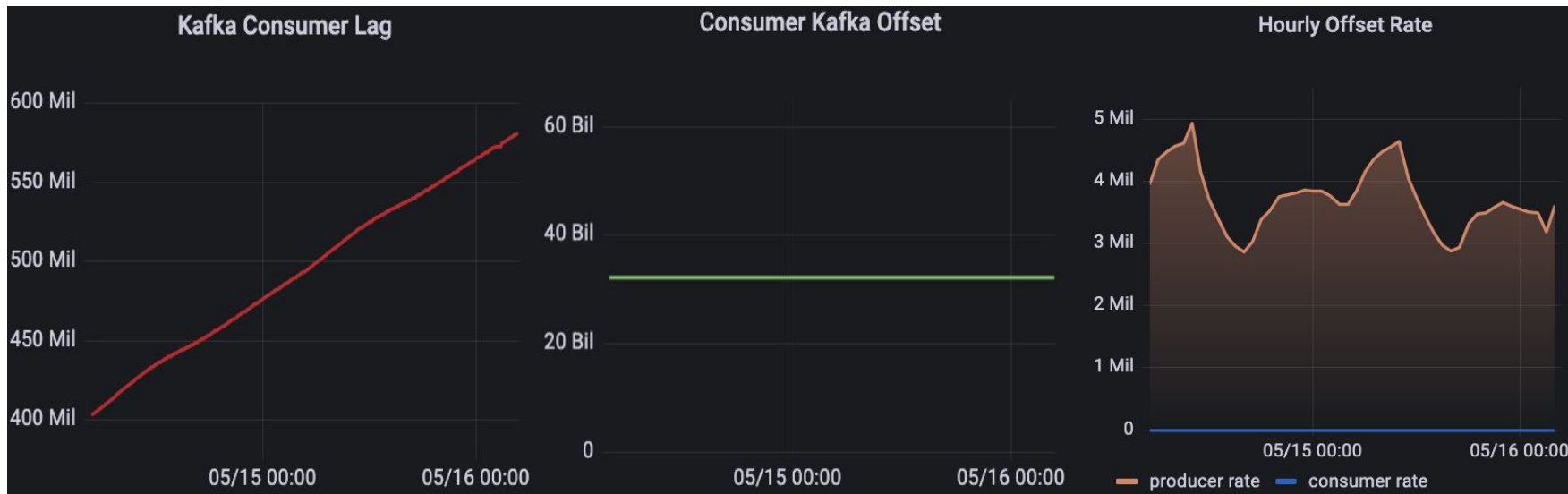
# Lag - Consumer is Slow!



Status of consumer: **WARNING**

Lag Series with Uptrend & Consumer Offset Series not Stalled

# Lag - Consumer Stalled

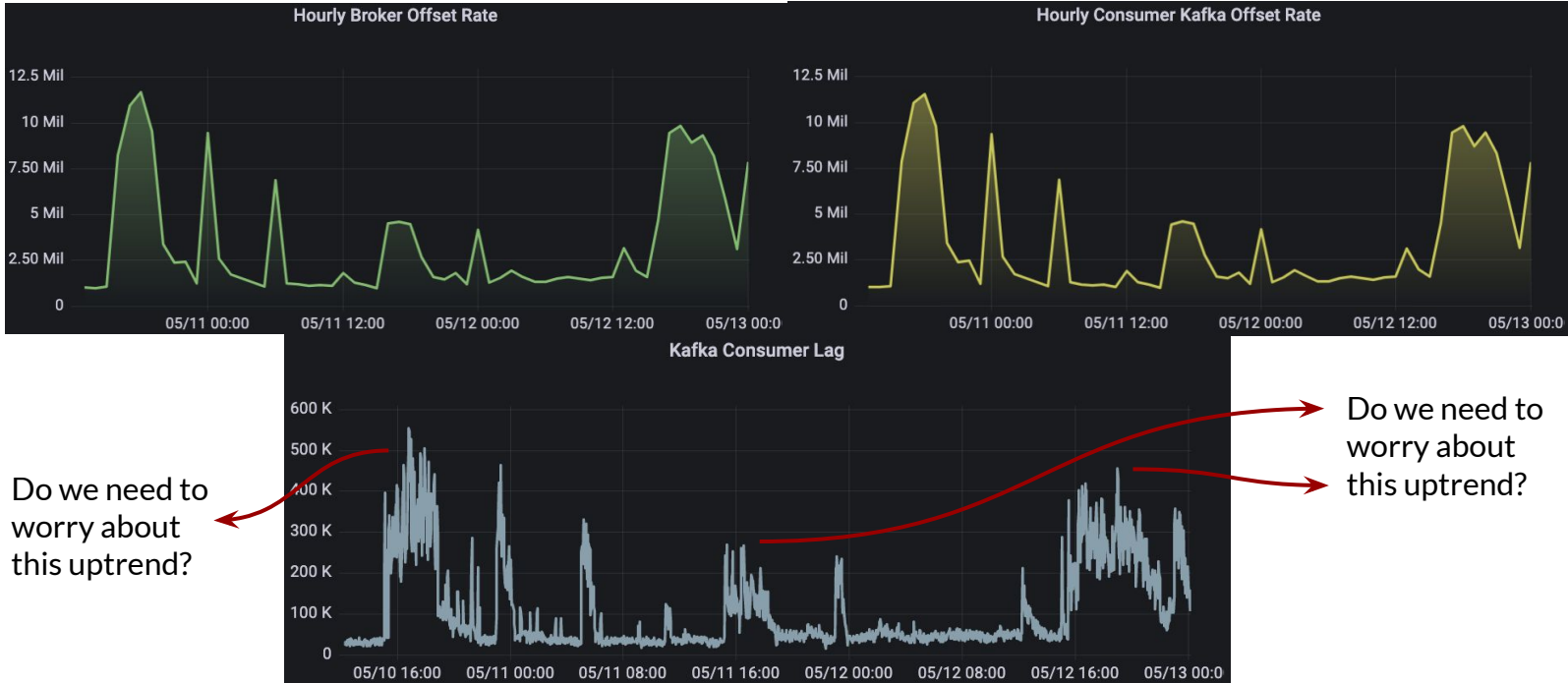


Status of consumer: **STALLED**

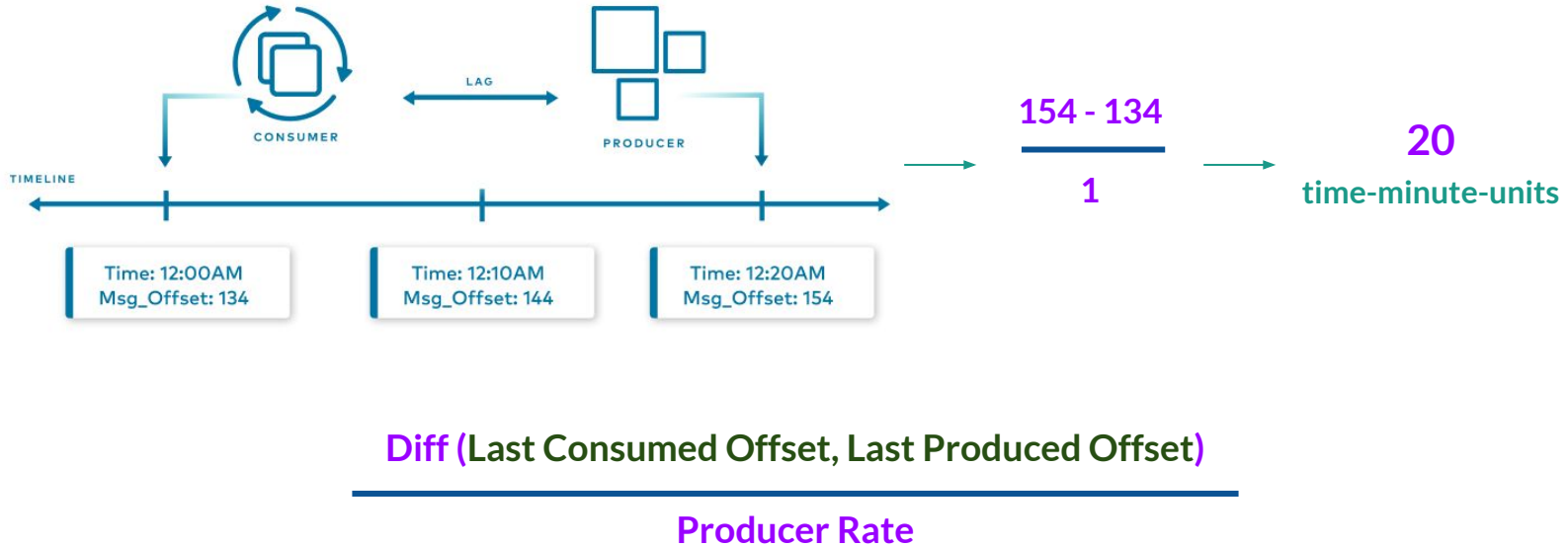
Lag Series with Uptrend & Consumer Offset Series Stalled



# Lag - Observability



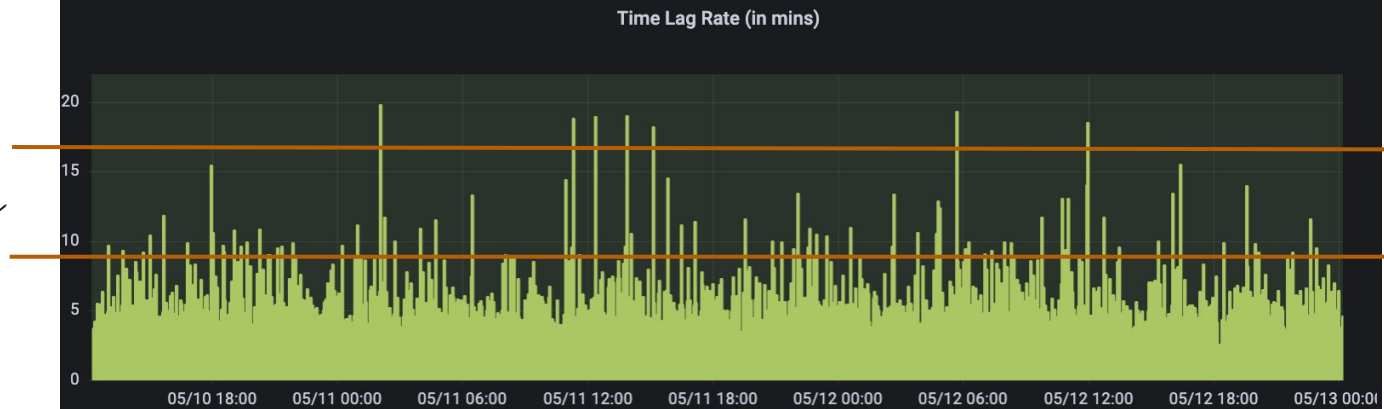
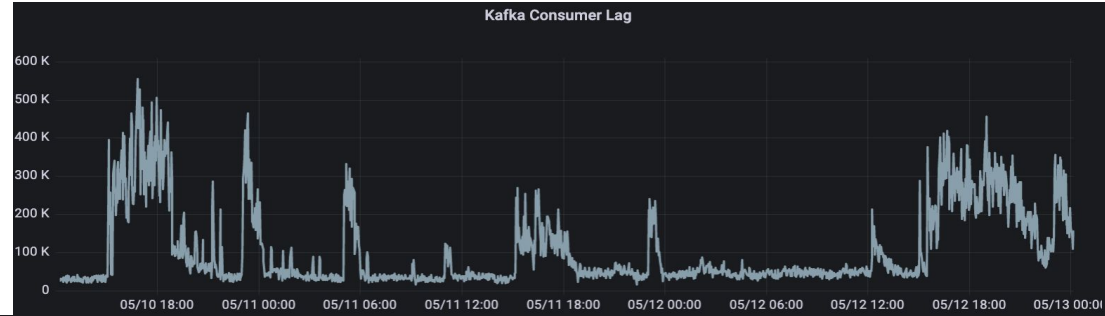
# Lag - Time Based



# Lag - Time Based



Normal Trend!



# Trend Analysis



Keep track of high-level metrics for:

- Rate of Topic Growth → Do we need more partitions?
- Weekly / Monthly / Periodic Producer / Consumer Rate → Keeping tabs on abnormal spikes!
- TTL / Retention data long enough to hold data for consumption → If time lag for consumer-partition goes beyond control!
- Infrastructure supporting Kafka cluster requirements → CPU, Memory, Network, IO capacity, GC Activity
- *Zookeeper supporting Kafka cluster requirements* → *How many topics / partitions state can be kept?*

\* ZK has been deprecated / removed in newer versions

# References





**Thank You!**  
**@sarkaramrit2**